.CSE 490/590 – Midterm Exam – March 12, 2025. V2 Solution

[Question 1] (10 Points)

Assume the presence of the following memories in a MIPS system:

i. L1 cache
ii. Main Memory
iii. L3 cache
iv. Registers
v. SSDs
vi. L2 cache
vii. Hard drive

Show the memory hierarchy and order them in terms of (State Increasing or decreasing order):

a. Speed
Registers -> L1 -> L2 -> L3 -> Main Memory -> SSDs -> Hard Drive (decreasing)

b. Cost per byte
Registers -> L1 -> L2 -> L3 -> Main Memory -> SSD -> Hard Drive (decreasing)

a. Memory capacity (Size)
Registers -> L1 -> L2 -> L3 -> Main Memory -> SSD -> Hard Drive (increasing)

[Question 2] (10 Points)

a) Consider a direct-mapped cache of size 4 bytes. Each index in the cache can hold only 1 word (here 1 word = 1 byte). Fill in the missing cache blocks at each step according to the address reference and specify whether it is a hit or a miss. The initial state of the cache is provided. Address references are (in order):

Reference: 0x4 0x6 0xC 0x6

Tag	Cache Content
10b	Mem(0x8)
01b	Mem(0x5)
10b	Mem(0xA)
11b	Mem(0xF)

Tag	Cache Content
01b	M(0x4)
01b	M(0x5)
10b	M(0xA)
11b	M(0xF)

hit / miss

Tag	Cache Content
01b	M(0x4)
01b	M(0x5)
01b	M(0x6)
11b	M(0xF)

hit / miss

Tag	Cache Content
11b	M(0xC)
01b	M(0x5)
01b	M(0x6)
11b	M(0xF)

hit / miss

Tag	Cache Content
11b	M(0xC)
01b	M(0x5)
01b	M(0x6)
11b	M(0xF)

hit / miss

b) Calculate the hit rate for part a.
¼ = 0.25

[Question 3] (12 Points)
Consider three different processors P1, P2, and P3 executing the same instruction set.

Processor	Clock Speed in GHz	CPI
P1	3	2.5
P2	1.5	1.5
P3	4.0	2.0

Which processor has the highest performance expressed in instructions per second?
(1 GHz = 10^9 Hz). **Show your work.**

IPS = Cycles per second/CPI

P1: 3/2.5 = 1.2
P2: 1.5/1.5 = 1
P3: 4/2 = 2

P3 is highest performance

[Question 4] (12 Points)
In the following instruction sequence, find all hazards. Rename the registers to eliminate the anti-dependencies and output dependences. Assume you have FP registers up to `$F16` available. **Show your work.**

```
div.s      F0, F2, F1
mult.s     F1, F0, F6
add.s      F0, F2, F12
sub.s      F8, F14, F2
```

Potential hazards
RAW: F0 -> div.s and mult.s;
WAR: F1 -> div.s and mult.s, F0 -> mult.s and add.s
WAW: F0 -> div.s and add.s

```
div.s      F0, F2, F1
mult.s     F9, F0, F6
add.s      F10, F2, F12
sub.s      F8, F14, F2
```

[Question 5] (12 Points)

Consider three levels of cache backed by DRAM:
- Access time for L1 cache is 2 cycles and miss rate is 60%
- Access time for L2 cache is 15 cycles and miss rate is 8%
- Access time for L3 cache is 60 cycles and miss rate is 4%
- Access time for DRAM is 110 cycles and miss rate is 0% (no misses)

Calculate Average Memory Access Time (AMAT). **Show your work.**
$AMAT = Average\ Memory\ Acess\ Time = Hit\ Time + Miss\ Rate * Miss\ Penalty$

$$\text{AMAT} = \text{HT\_1} + \text{MR\_1}(\text{HT\_2} + \text{MR\_2}(\text{HT\_3} + \text{MR\_3}(\text{HT\_4} + \text{MR\_4}(\text{MP\_4}))))$$
$$= 2 + 0.6(15 + 0.08(60 + 0.04(110 + 0(0))))$$
$$= 14.0912 \text{ cycles}$$

[Question 6] (12 Points)

Compare the execution time of the pipelined datapath with the non-pipelined datapath for the following instruction sequence:

```
lw $s1, 100($s0)
add $s2, $s3, $s0
sw $s4, 0($s5)
```

The following table provides time taken in each stage for the specific instructions:

Instruction	Instruction Fetch	Register Read	ALU Operation	Memory Access	Register Write	Total Time
lw	300ps	150ps	300ps	300ps	150ps	1200ps
sw	300ps	150ps	300ps	300ps		1050ps
R-format	300ps	150ps	300ps		150ps	900ps
beq	300ps	150ps	300ps			750ps

Non-pipelined:
In a non-pipelined processor, each instruction is executed sequentially, meaning the total execution time is the sum of individual instruction execution times.
Execution time of non-pipelined datapath = 1200 + 900 + 1050 = 3150 ps

Pipelined:
In a pipelined processor, multiple instructions overlap in execution. The execution time is determined by the number of cycles required to complete all instructions, assuming each stage takes 300 ps.
Total 7 cycles are required to execute given instructions.
Execution time of a pipelined datapath = 300 * 7 = 2100 ps

Execution time(pipelined) < Execution time(non-pipelined)
It shows that pipelining improved performance.

[Question 7] (16 Points)

Assume a 5-stage, static dual-issue MIPS processor that can perform data forwarding. The 2-issue MIPS helps us issue arithmetic and load/store instructions in parallel. The ideal 2-issue MIPS, with no data dependencies will have a pipelining diagram as follows:

Address	Instruction type	Pipeline Stages						
n	ALU/branch	IF	ID	EX	MEM	WB		
n + 4	Load/store	IF	ID	EX	MEM	WB		
n + 8	ALU/branch		IF	ID	EX	MEM	WB	
n + 12	Load/store		IF	ID	EX	MEM	WB	
n + 16	ALU/branch			IF	ID	EX	MEM	WB
n + 20	Load/store			IF	ID	EX	MEM	WB

For the instruction sequence given below, draw the pipelining diagram while taking all data dependencies into account. Show the data forwarding if necessary. Find the total number of clock cycles taken for the execution of the instruction sequence. An empty table for pipelining is given below for convenience

```
add  $s1, $s2, $s2
add  $s3, $s2, $s3
sw   $s1, 4($t2)
add  $s5, $s3, $s1
```

Address	**Instruction**	**Pipeline Stages**							
	add s1,s2,s2	IF	ID	EX	MEM	WB			
	add s3,s2,s3		IF	ID	EX	MEM	WB		
	sw s1, 4(t2)		IF	ID	EX	MEM	WB		
	add s5, s3,s1			IF	ID	EX	MEM	WB	

7 clock cycles

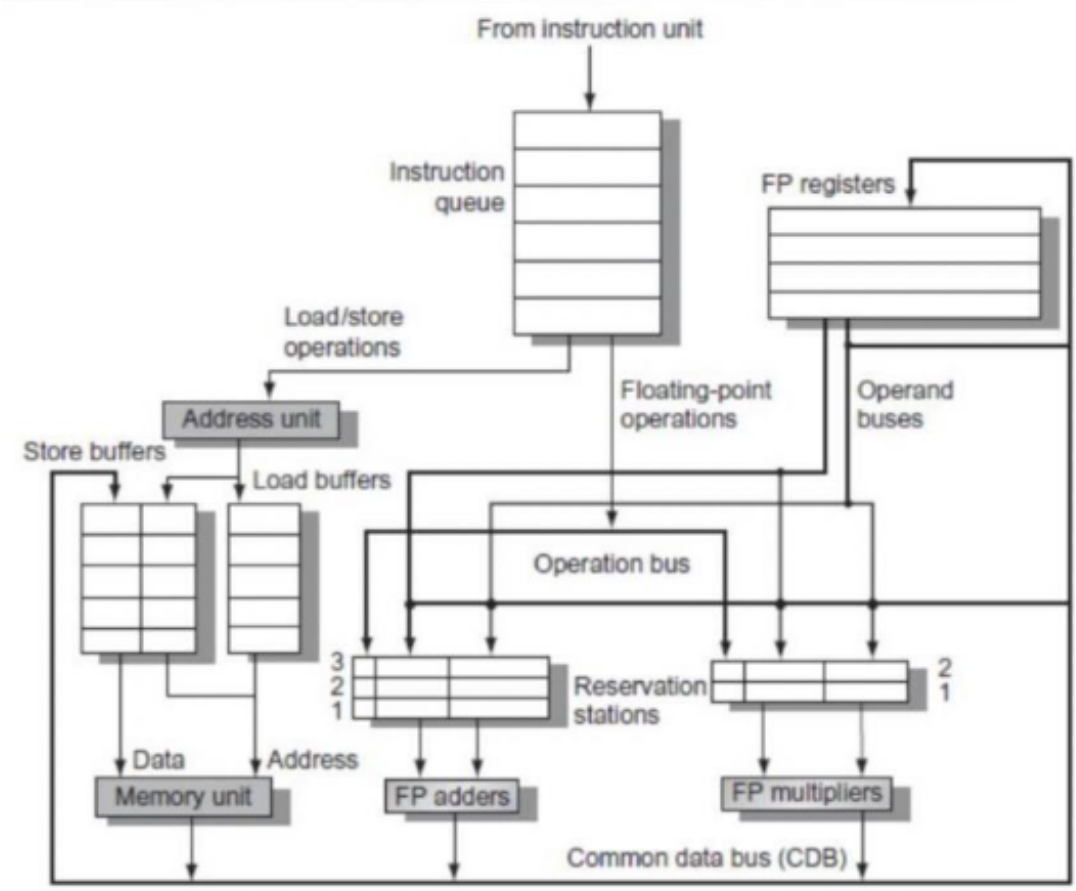
[Question 8] (16 Points)

Consider the following floating-point instruction sequence on a processor (shown below) which uses Tomasulo's Algorithm to dynamically schedule instructions (dual-issue per cycle – no speculation). The processor has the following non-pipelined execution units:

- A 2-cycle FP add unit
- A 3-cycle FP multiply unit

Assume instructions can begin to execute in the same cycle as soon as it is dispatched and resides in Reservation Stations. Trace the execution by showing the contents of Reservation Stations and FP Registers at the **start** of each cycle, after instructions have been issued for that cycle. Initial register values are given in the cycle 1 tables. As completed, result is placed in Common Data Bus, and is ready for use in the next cycle.

```
w:  add r0, r6, r4
x:  mul r4, r0, r2
y:  add r0, r2, r0
```



Also, circle the instruction being executed in the table below.

1

Adder Reservation Stations					
		Tag 1	S1	Tag 2	S2
1	w		2.0		6.0
2					
3					
Instruction Executed: w x y					

Multiply Reservation Stations					
		Tag 1	S1	Tag 2	S2
4	x	1			4.0
5					
Instruction Executed: w x y					

FP Registers			
	Busy	Tag	Data
0	Yes	1	3.0
2			4.0
4	Yes	4	2.0
6			6.0

2

Adder Reservation Stations					
		Tag 1	S1	Tag 2	S2
1	w		2.0		6.0
2	y		4 .0	1	
3					
Instruction Executed: w x y					

Multiply Reservation Stations					
		Tag 1	S1	Tag 2	S2
4	x	1			4.0
5					
Instruction Executed: w x y					

FP Registers			
	Busy	Tag	Data
0	Yes	2	3.0
2			4.0
4	Yes	4	2.0
6			6.0

3

Adder Reservation Stations					
		Tag 1	S1	Tag 2	S2
1					
2	y		4.0		8.0
3					
Instruction Executed: w x y					

Multiply Reservation Stations					
		Tag 1	S1	Tag 2	S2
4	x		8.0		4.0
5					
Instruction Executed: w x y					

FP Registers			
	Busy	Tag	Data
0	Yes	2	3.0
2			4.0
4	Yes	4	2.0
6			6.0

4

Adder Reservation Stations					
		Tag 1	S1	Tag 2	S2
1					
2	y		4.0		8.0
3					
Instruction Executed: w x y					

Multiply Reservation Stations					
		Tag 1	S1	Tag 2	S2
4	x		8.0		4.0
5					
Instruction Executed: w x y					

FP Registers			
	Busy	Tag	Data
0	Yes	2	3.0
2			4.0
4	Yes	4	2.0
6			6.0

5

Adder Reservation Stations					
		Tag 1	S1	Tag 2	S2
1					
2					
3					
Instruction Executed: w x y					

Multiply Reservation Stations					
		Tag 1	S1	Tag 2	S2
4	x		8.0		4.0
5					
Instruction Executed: w x y					

FP Registers			
	Busy	Tag	Data
0			12.0
2			4.0
4	Yes	4	2.0
6			6.0

6

Adder Reservation Stations					
		Tag 1	S1	Tag 2	S2
1					
2					
3					
Instruction Executed: w x y					

Multiply Reservation Stations					
		Tag 1	S1	Tag 2	S2
4					
5					
Instruction Executed: w x y					

FP Registers			
	Busy	Tag	Data
0			12.0
2			4.0
4			32.0
6			6.0

Rubric is the same as in V1